Adaptive Mixture Methods Based on Bregman Divergences

Mehmet A. Donmez [a], Huseyin A. Inan [a], Suleyman S. Kozat [a,*]

[a] *Department of Electrical and Computer Engineering, Koc University, Istanbul, Tel: 90 212 3501840.*

Abstract

We investigate adaptive mixture methods that linearly combine outputs of m constituent filters running in parallel to model a desired signal. We use "Bregman divergences" and obtain certain multiplicative updates to train the linear combination weights under an affine constraint or without any constraints. We use unnormalized relative entropy and relative entropy to define two different Bregman divergences that produce an unnormalized exponentiated gradient update and a normalized exponentiated gradient update on the mixture weights, respectively. We then carry out the mean and the mean-square transient analysis of these adaptive algorithms when they are used to combine outputs of m constituent filters. We illustrate the accuracy of our results and demonstrate the effectiveness of these updates for sparse mixture systems.

Key words: Adaptive mixture, Bregman divergence, affine mixture, multiplicative update.

* Corresponding author.
Email addresses: mdonmez@ku.edu.tr (Mehmet A. Donmez), huseyin.inan@boun.edu.tr (Huseyin A. Inan), skozat@ku.edu.tr (Suleyman S. Kozat).

1 Introduction

In this paper, we study adaptive mixture methods based on "Bregman divergences" [1,2] that combine outputs of m constituent filters running in parallel on the same task. The overall system has two stages [3–8]. The first stage contains adaptive filters running in parallel to model a desired signal. The outputs of these adaptive filters are then linearly combined to produce the final output of the overall system in the second stage. We use Bregman divergences and obtain certain multiplicative updates [9], [2], [10] to train these linear combination weights under an affine constraint [11] or without any constraints [12]. We use unnormalized [2] and normalized relative entropy [9] to define two different Bregman divergences that produce the unnormalized exponentiated gradient update (EGU) and the exponentiated gradient update (EG) on the mixture weights [9], respectively. We then perform the mean and the mean-square transient analysis of these adaptive mixtures when they are used to combine outputs of m constituent filters. We emphasize that to the best of our knowledge, this is the first mean and mean-square transient analysis of the EGU algorithm and the EG algorithm in the mixture framework (which naturally covers the classical framework also [13, 14]). We illustrate the accuracy of our results through simulations in different configurations and demonstrate advantages of the introduced algorithms for sparse mixture systems.

Adaptive mixture methods are utilized in a wide range of signal processing applications in order to improve the steady-state and/or convergence performance over the constituent filters [11,12,15]. An adaptive convexly constrained mixture of two filters is studied in [15], where the convex combination is shown to be "universal" such that the combination performs at least as well as its best constituent filter in the steady-state [15]. The transient analysis of this adaptive convex combination is studied in [16], where the time evolution of the mean and variance of the mixture weights is provided. In similar lines,

an affinely constrained mixture of adaptive filters using a stochastic gradient update is introduced in [11]. The steady-state mean square error (MSE) of this affinely constrained mixture is shown to outperform the steady-state MSE of the best constituent filter in the mixture under certain conditions [11]. The transient analysis of this affinely constrained mixture for m constituent filters is carried out in [17]. The general linear mixture framework as well as the steady-state performances of different mixture configurations are studied in [12].

In this paper, we use Bregman divergences to derive multiplicative updates on the mixture weights. We use the unnormalized relative entropy and the relative entropy as distance measures and obtain the EGU algorithm and the EG algorithm to update the combination weights under an affine constraint or without any constraints. We then carry out the mean and the mean-square transient analysis of these adaptive mixtures when they are used to combine m constituent filters. We point out that the EG algorithm is widely used in sequential learning theory [18] and minimizes an approximate final estimation error while penalizing the distance between the new and the old filter weights. In network and acoustic echo cancellation applications, the EG algorithm is shown to converge faster than the LMS algorithm [14, 19] when the system impulse response is sparse [13]. Similarly, in our simulations, we observe that using the EG algorithm to train the mixture weights yields increased convergence speed compared to using the LMS algorithm to train the mixture weights [11, 12] when the combination favors only a few of the constituent filters in the steady state, i.e., when the final steady-state combination vector is sparse. We also observe that the EGU algorithm and the LMS algorithm show similar performance when they are used to train the mixture weights even if the final steady-state mixture is sparse.

To summarize, the main contributions of this paper are as follows:

- We use Bregman divergences to derive multiplicative updates on affinely constrained and unconstrained mixture weights adaptively combining outputs of m constituent filters.
- We use the unnormalized relative entropy and the relative entropy to define two different Bregman divergences that produce the EGU algorithm and the EG algorithm to update the affinely constrained and unconstrained mixture weights.
- We perform the mean and the mean-square transient analysis of the affinely constrained and unconstrained mixtures using the EGU algorithm and the EG algorithm.

The organization of the paper is as follows. In Section II, we first describe the mixture framework. In Section III, we study the affinely constrained and unconstrained mixture methods updated with the EGU algorithm and the EG algorithm. In Section IV, we first perform the transient analysis of the affinely constrained mixtures and then continue with the transient analysis of the unconstrained mixtures. Finally, in Section V, we perform simulations to show the accuracy of our results and to compare performances of the different adaptive mixture methods. The paper concludes with certain remarks in Section VI.

2 System Description

2.1 Notation

In this paper, all vectors are column vectors and represented by boldface lowercase letters. Matrices are represented by boldface capital letters. For presentation purposes, we work only with real data. Given a vector $\boldsymbol{w}$, $w^{(i)}$ denotes the ith individual entry of $\boldsymbol{w}$, $\boldsymbol{w}^T$ is the transpose of $\boldsymbol{w}$, $\|\boldsymbol{w}\|_1 \triangleq$

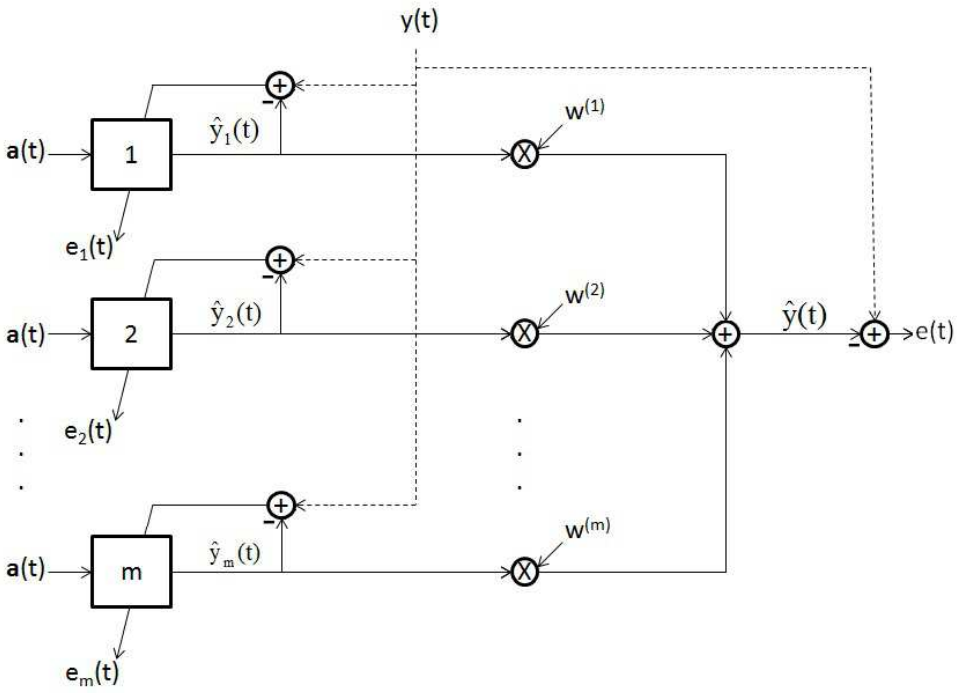


Fig. 1. A linear mixture of outputs of m adaptive filters.

$\sum_i |w^{(i)}|$ is the l_1 norm; $\|\boldsymbol{w}\| \triangleq \sqrt{\boldsymbol{w}^T\boldsymbol{w}}$ is the l_2 norm. For a matrix $\boldsymbol{W}$, $\mathrm{tr}(\boldsymbol{W})$ is the trace. For a vector $\boldsymbol{w}$, $\mathrm{diag}(\boldsymbol{w})$ represents a diagonal matrix formed using the entries of $\boldsymbol{w}$. For a matrix $\boldsymbol{W}$, $\mathrm{diag}(\boldsymbol{W})$ represents a column vector that contains the diagonal entries of $\boldsymbol{W}$. For two vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, we define the concatenation $[\boldsymbol{v}_1; \boldsymbol{v}_2] \triangleq [\boldsymbol{v}_1^T\ \boldsymbol{v}_2^T]^T$. For a random variable v, $\bar{v}$ is the expected value. For a random vector $\boldsymbol{v}$ (or a random matrix $\boldsymbol{V}$), $\bar{\boldsymbol{v}}$ (or $\bar{\boldsymbol{V}}$) represents the expected value of each entry. Vectors (or matrices) $\mathbf{1}$ and $\mathbf{0}$, with an abuse of notation, denote vectors (or matrices) of all ones or zeros, respectively, where the size of the vector (or the matrix) is understood from the context.

2.2 System Description

The framework that we study has two stages. In the first stage, we have m adaptive filters producing outputs $\hat{y}_i(t)$, $i = 1, \ldots, m$, running in parallel to

model a desired signal $y(t)$ as seen in Fig. 1. The second stage is the mixture stage, where the outputs of the first stage filters are combined to improve the steady-state and/or the transient performance over the constituent filters. We linearly combine the outputs of the first stage filters to produce the final output as $\hat{y}(t) = \boldsymbol{w}^T(t)\boldsymbol{x}(t)$, where $\boldsymbol{x}(t) \stackrel{\triangle}{=} [\hat{y}_1(t), \ldots, \hat{y}_m(t)]^T$ and train the mixture weights using multiplicative updates (or exponentiated gradient updates) [2]. We point out that in order to satisfy the constraints and derive the multiplicative updates [9], [20], we use reparametrization of the mixture weights as $\boldsymbol{w}(t) = \boldsymbol{f}(\boldsymbol{z}(t))$ and perform the update on $\boldsymbol{z}(t)$ as

$$\boldsymbol{z}(t+1) = \arg\min_{\boldsymbol{z}} \left\{ d(\boldsymbol{z}, \boldsymbol{z}(t)) + \mu \, l\Big(y(t), \boldsymbol{f}^T(\boldsymbol{z})\boldsymbol{x}(t)\Big) \right\}, \tag{1}$$

where μ is the learning rate of the update, $d(\cdot,\cdot)$ is an appropriate distance measure and $l(\cdot,\cdot)$ is the instantaneous loss. We emphasize that in (1), the updated vector $\boldsymbol{z}$ is forced to be close to the present vector $\boldsymbol{z}(t)$ by $d(\boldsymbol{z}(t+1), \boldsymbol{z}(t))$, while trying to accurately model the current data by $l\Big(y(t), \boldsymbol{f}^T(\boldsymbol{z})\boldsymbol{x}(t)\Big)$. However, instead of directly minimizing (1), a linearized version of (1)

$$\begin{aligned} \boldsymbol{z}(t+1) = \arg\min_{\boldsymbol{z}} \Bigg\{ & d(\boldsymbol{z}, \boldsymbol{z}(t)) + l\left(y(t), \boldsymbol{f}^T(\boldsymbol{z}(t))\boldsymbol{x}(t)\right) \\ & + \mu \nabla_{\boldsymbol{z}} l\left(y(t), \boldsymbol{f}^T(\boldsymbol{z})\boldsymbol{x}(t)\right)^T \Big|_{\boldsymbol{z}=\boldsymbol{z}(t)} (\boldsymbol{z} - \boldsymbol{z}(t)) \Bigg\} \end{aligned} \tag{2}$$

is minimized to get the desired update. As an example, if we use the l_2-norm as the distance measure, i.e., $d(\boldsymbol{z}, \boldsymbol{z}(t)) = \|\boldsymbol{z} - \boldsymbol{z}(t)\|^2$, and the square error as the instantaneous loss, i.e., $l\Big(y(t), \boldsymbol{f}^T(\boldsymbol{z})\boldsymbol{x}(t)\Big) = [y(t) - \boldsymbol{f}^T(\boldsymbol{z})\boldsymbol{x}(t)]^2$ with $\boldsymbol{f}(\boldsymbol{z}) = \boldsymbol{z}$, then we get the stochastic gradient update on $\boldsymbol{w}(t)$, i.e.,

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \mu e(t)\boldsymbol{x}(t),$$

in (2).

In the next section, we use the unnormalized relative entropy

$$d_1(\boldsymbol{z}, \boldsymbol{z}(t)) = \left\{ \sum_{i=1}^{m} \left[z^{(i)} \ln \left(\frac{z^{(i)}}{z^{(i)}(t)} \right) + z^{(i)}(t) - z^{(i)} \right] \right\} \tag{3}$$

for positively constrained $\boldsymbol{z}$ and $\boldsymbol{z}(t)$, $\boldsymbol{z} \in \mathbb{R}_+^m$, $\boldsymbol{z}(t) \in \mathbb{R}_+^m$, and the relative entropy

$$d_2(\boldsymbol{z}, \boldsymbol{z}(t)) = \left\{ \sum_{i=1}^{m} \left[z^{(i)} \ln \left(\frac{z^{(i)}}{z^{(i)}(t)} \right) \right] \right\}, \tag{4}$$

where $\boldsymbol{z}$ is constrained to be in an extended simplex such that $z^{(i)} \geq 0$, $\sum_{k=1}^{m} z^{(i)} = u$ for some $u \geq 1$ as the distance measures, with appropriately selected $\boldsymbol{f}(\cdot)$ to derive updates on mixture weights under different constraints. We first investigate affinely constrained mixture of m adaptive filters, and then continue with the unconstrained mixture using (3) and (4) as the distance measures.

3 Adaptive Mixture Algorithms

In this section, we investigate affinely constrained and unconstrained mixtures updated with the EGU algorithm and the EG algorithm.

3.1 Affinely Constrained Mixture

When an affine constraint is imposed on the mixture such that $\boldsymbol{w}^T(t)\mathbf{1} = 1$, we get

$$\begin{aligned}
&\hat{y}(t) = \boldsymbol{w}(t)^T \boldsymbol{x}(t), \\
&e(t) = y(t) - \hat{y}(t), \\
&w^{(i)}(t) = \lambda^{(i)}(t), \;\; i = 1, \ldots, m-1, \\
&w^{(m)}(t) = 1 - \sum_{i=1}^{m-1} \lambda^{(i)}(t),
\end{aligned}$$

where the $m-1$ dimensional vector $\boldsymbol{\lambda}(t) \stackrel{\triangle}{=} [\lambda^{(1)}(t), \ldots, \lambda^{(m-1)}(t)]^T$ is the unconstrained weight vector, i.e., $\boldsymbol{\lambda}(t) \in \mathbb{R}^{m-1}$. Using $\boldsymbol{\lambda}(t)$ as the unconstrained weight vector, the error can be written as $e(t) = \left[y(t) - \hat{y}_m(t) \right] - \boldsymbol{\lambda}^T(t)\boldsymbol{\delta}(t)$, where $\boldsymbol{\delta}(t) \stackrel{\triangle}{=} [\hat{y}_1(t) - \hat{y}_m(t), \ldots, \hat{y}_{m-1}(t) - \hat{y}_m(t)]^T$. To be able to derive a

multiplicative update on $\boldsymbol{\lambda}(t)$, we use

$$\boldsymbol{\lambda}(t) = \boldsymbol{\lambda}_1(t) - \boldsymbol{\lambda}_2(t),$$

where $\boldsymbol{\lambda}_1(t)$ and $\boldsymbol{\lambda}_2(t)$ are constrained to be nonnegative, i.e., $\boldsymbol{\lambda}_i(t) \in \mathbb{R}_+^{m-1}$, $i = 1, 2$. After we collect unconstrained weights in $\boldsymbol{\lambda}_a(t) = [\boldsymbol{\lambda}_1(t); \boldsymbol{\lambda}_2(t)]$, we define a function of loss $e(t)$ as

$$l_{\mathrm{a}}\left(\boldsymbol{\lambda}_a(t)\right) \triangleq e^2(t)$$

and update positively constrained $\boldsymbol{\lambda}_a(t)$ as follows.

3.1.1 Unnormalized Relative Entropy

Using the unconstrained relative entropy as the distance measure, we get

$$\begin{aligned}\boldsymbol{\lambda}_a(t+1) = \arg\min_{\boldsymbol{\lambda}} \Bigg\{ \sum_{i=1}^{2(m-1)} \left[\lambda^{(i)} \ln \left(\frac{\lambda^{(i)}}{\lambda_a^{(i)}(t)} \right) + \lambda_a^{(i)}(t) - \lambda^{(i)} \right] + \\ \mu \left[l_{\mathrm{a}}\left(\boldsymbol{\lambda}_a(t)\right) + \nabla_{\boldsymbol{\lambda}} l_{\mathrm{a}}\left(\boldsymbol{\lambda}\right)^T \big|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}_a(t)} \left(\boldsymbol{\lambda} - \boldsymbol{\lambda}_a(t)\right) \right] \Bigg\}.\end{aligned}$$

After some algebra this yields

$$\begin{aligned}\lambda_a^{(i)}(t+1) &= \lambda_a^{(i)}(t) \exp\left\{ \mu e(t) (\hat{y}_i(t) - \hat{y}_m(t)) \right\}, i = 1, \ldots, m-1, \\ \lambda_a^{(i)}(t+1) &= \lambda_a^{(i)}(t) \exp\left\{ -\mu e(t) (\hat{y}_i(t) - \hat{y}_m(t)) \right\}, i = m, \ldots, 2(m-1),\end{aligned}$$

providing the multiplicative updates on $\boldsymbol{\lambda}_1(t)$ and $\boldsymbol{\lambda}_2(t)$.

3.1.2 Relative Entropy

Using the relative entropy as the distance measure, we get

$$\begin{aligned}\boldsymbol{\lambda}_a(t+1) = \arg\min_{\boldsymbol{\lambda}} \Bigg\{ \sum_{i=1}^{2(m-1)} \left[\lambda^{(i)} \ln \left(\frac{\lambda^{(i)}}{\lambda_a^{(i)}(t)} \right) + \gamma(u - \mathbf{1}^T \boldsymbol{\lambda}) \right] + \\ \mu \left[l_{\mathrm{a}}\left(\boldsymbol{\lambda}_a(t)\right) + \nabla_{\boldsymbol{\lambda}} l_{\mathrm{a}}\left(\boldsymbol{\lambda}\right)^T \big|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}_a(t)} \left(\boldsymbol{\lambda} - \boldsymbol{\lambda}_a(t)\right) \right] \Bigg\},\end{aligned}$$

where γ is the Lagrange multiplier. This yields

$$\lambda_a^{(i)}(t+1) = u\frac{\lambda_a^{(i)}(t)\exp\left\{\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\right\}}{\sum_{k=1}^{m-1}\left[\lambda_a^{(k)}(t)\exp\left\{\mu e(t)(\hat{y}_k(t)-\hat{y}_m(t))\right\}+\lambda_a^{(k+m-1)}(t)\exp\left\{-\mu e(t)(\hat{y}_k(t)-\hat{y}_m(t))\right\}\right]},$$

$i = 1, \ldots, m-1,$

$$\lambda_a^{(i)}(t+1) = u\frac{\lambda_a^{(i)}(t)\exp\left\{-\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\right\}}{\sum_{k=1}^{m-1}\left[\lambda_a^{(k)}(t)\exp\left\{\mu e(t)(\hat{y}_k(t)-\hat{y}_m(t))\right\}+\lambda_a^{(k+m-1)}(t)\exp\left\{-\mu e(t)(\hat{y}_k(t)-\hat{y}_m(t))\right\}\right]},$$

$i = m, \ldots, 2(m-1),$

providing the multiplicative updates on $\boldsymbol{\lambda}_a(t)$.

3.2 Unconstrained Mixture

Without any constraints on the combination weights, the mixture stage can be written as

$$\hat{y}(t) = \boldsymbol{w}^T(t)\boldsymbol{x}(t),$$
$$e(t) = y(t) - \hat{y}(t),$$

where $\boldsymbol{w}(t) \in \mathbb{R}^m$. To be able to derive a multiplicative update, we use a change of variables,

$$\boldsymbol{w}(t) = \boldsymbol{w}_1(t) - \boldsymbol{w}_2(t),$$

where $\boldsymbol{w}_1(t)$ and $\boldsymbol{w}_2(t)$ are constrained to be nonnegative, i.e., $\boldsymbol{w}_i(t) \in \mathbb{R}_+^m$, $i = 1, 2$. We then collect the unconstrained weights $\boldsymbol{w}_a(t) = [\boldsymbol{w}_1(t); \boldsymbol{w}_2(t)]$ and define a function of the loss $e(t)$ as

$$l_{\mathrm{U}}\left(\boldsymbol{w}_a(t)\right) \stackrel{\triangle}{=} e^2(t).$$

3.2.1 Unnormalized Relative Entropy

Defining cost function similar to (4) and minimizing it with respect to $\boldsymbol{w}$ yields

$$w_a^{(i)}(t+1) = w_a^{(i)}(t)\exp\left\{\mu e(t)\hat{y}_i(t)\right\}, i = 1, \ldots, m,$$
$$w_a^{(i)}(t+1) = w_a^{(i)}(t)\exp\left\{-\mu e(t)\hat{y}_i(t)\right\}, i = m+1, \ldots, 2m,$$

providing the multiplicative update on $\boldsymbol{w}_a(t)$.

3.2.2 Relative Entropy

Using the relative entropy under the simplex constraint on $\boldsymbol{w}$, we get the updates

$$w_a^{(i)}(t+1) = u\frac{w_a^{(i)}(t)\exp\{\mu e(t)\hat{y}_i(t)\}}{\sum_{k=1}^{m}\left[w_a^{(k)}(t)\exp\{\mu e(t)\hat{y}_k(t)\} + w_a^{(k+m)}(t)\exp\{-\mu e(t)\hat{y}_k(t)\}\right]},$$

$i = 1, \ldots, m,$

$$w_a^{(i)}(t+1) = u\frac{w_a^{(i)}(t)\exp\{-\mu e(t)\hat{y}_i(t)\}}{\sum_{k=1}^{m}\left[w_a^{(k)}(t)\exp\{\mu e(t)\hat{y}_k(t)\} + w_a^{(k+m)}(t)\exp\{-\mu e(t)\hat{y}_k(t)\}\right]},$$

$i = m+1 \ldots, 2m.$

In the next section, we study the transient analysis of these four adaptive mixture algorithms.

4 Transient Analysis

In this section, we study the mean and the mean-square transient analysis of the adaptive mixture methods. We start with the affinely constrained combination.

4.1 Affinely Constrained Mixture

We first perform the transient analysis of the mixture weights updated with the EGU algorithm. Then, we continue with the transient analysis of the mixture weights updated with the EG algorithm.

4.1.1 Unconstrained Relative Entropy

For the affinely constrained mixture updated with the EGU algorithm, we have the multiplicative update as

$$\begin{aligned}\lambda_1^{(i)}(t+1) &= \lambda_1^{(i)}(t)\exp\left\{\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\right\},\\ &= \lambda_1^{(i)}(t)\sum_{k=0}^{\infty}\frac{\Big(\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\Big)^k}{k!}, \qquad (5)\\ \lambda_2^{(i)}(t+1) &= \lambda_2^{(i)}(t)\exp\left\{-\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\right\},\\ &= \lambda_2^{(i)}(t)\sum_{k=0}^{\infty}\frac{\Big(-\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\Big)^k}{k!}, \qquad (6)\end{aligned}$$

for $i = 1, \ldots, m-1$. If $e(t)$ and $\hat{y}_i(t) - \hat{y}_m(t)$ for each $i = 1, \ldots, m-1$ are bounded, then we can write (5) and (6) as

$$\lambda_1^{(i)}(t+1) = \lambda_1^{(i)}(t)\Big(1+\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t)) + O(\mu^2)\Big), \qquad (7)$$

$$\lambda_2^{(i)}(t+1) = \lambda_2^{(i)}(t)\Big(1-\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t)) + O(\mu^2)\Big), \qquad (8)$$

for $i = 1, \ldots, m-1$. Since μ is usually relatively small [2], we approximate (7) and (8) as

$$\lambda_1^{(i)}(t+1) = \lambda_1^{(i)}(t)\Big(1+\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\Big), \qquad (9)$$

$$\lambda_2^{(i)}(t+1) = \lambda_2^{(i)}(t)\Big(1-\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\Big). \qquad (10)$$

In our simulations, we illustrate the accuracy of the approximations (9) and (10) under the mixture framework. Using (9) and (10), we can obtain updates on $\boldsymbol{\lambda}_1(t)$ and $\boldsymbol{\lambda}_2(t)$ as

$$\boldsymbol{\lambda}_1(t+1) = \Big(I+\mu e(t)\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\Big)\boldsymbol{\lambda}_1(t), \qquad (11)$$

$$\boldsymbol{\lambda}_2(t+1) = \Big(I-\mu e(t)\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\Big)\boldsymbol{\lambda}_2(t). \qquad (12)$$

Collecting the weights in $\boldsymbol{\lambda}_a(t) = [\boldsymbol{\lambda}_1(t); \boldsymbol{\lambda}_2(t)]$, using the updates (11) and (12), we can write update on $\boldsymbol{\lambda}_a(t)$ as

$$\boldsymbol{\lambda}_a(t+1) = \Big(I+\mu e(t)\mathrm{diag}\Big(\boldsymbol{u}(t)\Big)\Big)\boldsymbol{\lambda}_a(t), \qquad (13)$$

where $\boldsymbol{u}(t)$ is defined as $\boldsymbol{u}(t) \triangleq [\boldsymbol{\delta}(t); -\boldsymbol{\delta}(t)]$.

For the desired signal $y(t)$, we can write $y(t)-\hat{y}_m(t) = \boldsymbol{\lambda}_0^T(t)\boldsymbol{\delta}(t)+e_0(t)$, where $\boldsymbol{\lambda}_0(t)$ is the optimum MSE solution at time t such that $\boldsymbol{\lambda}_0(t) \triangleq \boldsymbol{R}^{-1}(t)\boldsymbol{p}(t)$, $\boldsymbol{R}(t) \triangleq E\Big[\boldsymbol{\delta}(t)\boldsymbol{\delta}^T(t)\Big]$, $\boldsymbol{p}(t) \triangleq E\left\{\boldsymbol{\delta}(t)\Big[y(t)-\hat{y}_m(t)\Big]\right\}$ and $e_0(t)$ is zero-mean and uncorrelated with $\boldsymbol{\delta}(t)$. We next show that the mixture weights converge to the optimum solution in the steady-state such that $\lim_{t\to\infty} E\Big[\boldsymbol{\lambda}(t)\Big] = \lim_{t\to\infty} \boldsymbol{\lambda}_0(t)$ for properly selected μ.

Subtracting (12) from (11), we obtain

$$\begin{aligned}\boldsymbol{\lambda}(t+1) &= \boldsymbol{\lambda}(t) + \mu e(t)\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\Big(\boldsymbol{\lambda}_1(t)+\boldsymbol{\lambda}_2(t)\Big),\\ &= \boldsymbol{\lambda}(t) - \mu e(t)\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\boldsymbol{\lambda}(t) + 2\mu e(t)\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\boldsymbol{\lambda}_1(t). \qquad (14)\end{aligned}$$

Defining $\boldsymbol{\varepsilon}(t) \triangleq \boldsymbol{\lambda}_0(t) - \boldsymbol{\lambda}(t)$ and using $e(t) = \boldsymbol{\delta}^T(t)\boldsymbol{\varepsilon}(t) + e_0(t)$ in (14) yield

$$\begin{aligned}&\boldsymbol{\lambda}(t+1) = \boldsymbol{\lambda}(t) - \mu\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\boldsymbol{\lambda}(t)\boldsymbol{\delta}^T(t)\boldsymbol{\varepsilon}(t) - \mu\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\boldsymbol{\lambda}(t)e_0(t)\\ &+ 2\mu\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\boldsymbol{\lambda}_1(t)\boldsymbol{\delta}^T(t)\boldsymbol{\varepsilon}(t) + 2\mu\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\boldsymbol{\lambda}_1(t)e_0(t). \qquad (15)\end{aligned}$$

In (15), subtracting both sides from $\boldsymbol{\lambda}_0(t+1)$, we have

$$\begin{aligned}&\boldsymbol{\varepsilon}(t+1) = \boldsymbol{\varepsilon}(t) + \mu\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\boldsymbol{\lambda}(t)\boldsymbol{\delta}^T(t)\boldsymbol{\varepsilon}(t) + \mu\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\boldsymbol{\lambda}(t)e_0(t)\\ &- 2\mu\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\boldsymbol{\lambda}_1(t)\boldsymbol{\delta}^T(t)\boldsymbol{\varepsilon}(t) - 2\mu\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\boldsymbol{\lambda}_1(t)e_0(t)\\ &+ \Big[\boldsymbol{\lambda}_0(t+1) - \boldsymbol{\lambda}_0(t)\Big]. \qquad (16)\end{aligned}$$

Taking expectation of both sides of (16) and using

$$\begin{aligned}E\Big[\mu\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\boldsymbol{\lambda}(t)e_0(t)\Big] &= E\Big[\mu\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\boldsymbol{\lambda}(t)\Big]E[e_0(t)] = 0,\\ E\Big[2\mu\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\boldsymbol{\lambda}_1(t)e_0(t)\Big] &= E\Big[2\mu\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\boldsymbol{\lambda}_1(t)\Big]E[e_0(t)] = 0,\end{aligned}$$

and assuming that $\boldsymbol{\lambda}_1(t)$ and $\boldsymbol{\lambda}_2(t)$ are independent of $\boldsymbol{\varepsilon}(t)$ [17] yield

$$\begin{aligned}&E\Big[\boldsymbol{\varepsilon}(t+1)\Big] = E\Big[I - \mu\mathrm{diag}\Big(\boldsymbol{\lambda}_1(t)+\boldsymbol{\lambda}_2(t)\Big)\boldsymbol{\delta}(t)\boldsymbol{\delta}^T(t)\Big]E\Big[\boldsymbol{\varepsilon}(t)\Big]\\ &+ E\Big[\boldsymbol{\lambda}_0(t+1) - \boldsymbol{\lambda}_0(t)\Big]. \qquad (17)\end{aligned}$$

Assuming convergence of $\boldsymbol{R}(t)$ and $\boldsymbol{p}(t)$ (which is true for a wide range of adaptive methods in the first stage [16], [14, 21]), we obtain $\lim_{t\to\infty} E\Big[\boldsymbol{\lambda}_0(t+1)-\boldsymbol{\lambda}_0(t)\Big]=0$. If μ is chosen such that the eigenvalues of $E\Big[I-\mu\text{diag}\Big(\boldsymbol{\lambda}_1(t)+\boldsymbol{\lambda}_2(t)\Big)\boldsymbol{\delta}(t)\boldsymbol{\delta}^T(t)\Big]$ have strictly less than unit magnitude for every t, then $\lim_{t\to\infty} E\Big[\boldsymbol{\lambda}(t)\Big]=\lim_{t\to\infty}\boldsymbol{\lambda}_0(t)$.

For the transient analysis of the MSE, we have

$$\begin{aligned}
E[e^2(t)] &= E\left\{\Big[y(t)-\hat{y}_m(t)\Big]^2\right\} - 2\bar{\boldsymbol{\lambda}}_a^T(t)E\left\{\Big[y(t)-\hat{y}_m(t)\Big][\boldsymbol{\delta}(t);-\boldsymbol{\delta}(t)]\right\} \\
&\quad + E\left\{\boldsymbol{\lambda}_a^T(t)[\boldsymbol{\delta}(t);-\boldsymbol{\delta}(t)][\boldsymbol{\delta}(t);-\boldsymbol{\delta}(t)]^T\boldsymbol{\lambda}_a(t)\right\}, \\
&= E\left\{\Big[y(t)-\hat{y}_m(t)\Big]^2\right\} - 2\bar{\boldsymbol{\lambda}}_a^T(t)E\left\{\Big[y(t)-\hat{y}_m(t)\Big]\boldsymbol{u}(t)\right\} \\
&\quad + \text{tr}\Big(E\left[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\right]E\left\{\boldsymbol{u}(t)\boldsymbol{u}(t)^T\right\}\Big), \\
&= E\left\{\Big[y(t)-\hat{y}_m(t)\Big]^2\right\} - 2\bar{\boldsymbol{\lambda}}_a^T(t)\boldsymbol{\gamma}(t) + \text{tr}\Big(E\left[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\right]\boldsymbol{\Gamma}(t)\Big),
\end{aligned} \tag{18}$$

where we define $\boldsymbol{\gamma}(t)\triangleq E\left\{\boldsymbol{u}(t)\Big[y(t)-\hat{y}_m(t)\Big]\right\}$ and $\boldsymbol{\Gamma}(t)\triangleq E\Big[\boldsymbol{u}(t)\boldsymbol{u}^T(t)\Big]$.

For the recursion of $\bar{\boldsymbol{\lambda}}_a(t)=E[\boldsymbol{\lambda}_a(t)]$, using (13), we get

$$\bar{\boldsymbol{\lambda}}_a(t+1)=\bar{\boldsymbol{\lambda}}_a(t)+\mu\text{diag}\Big(\boldsymbol{\gamma}(t)\Big)\bar{\boldsymbol{\lambda}}_a(t)-\mu\text{diag}\Big(E[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)]\boldsymbol{\Gamma}(t)\Big). \tag{19}$$

Using (32), assuming $\boldsymbol{\lambda}_a(t)$ is Gaussian and assuming $\lambda_a^{(i)}(t)$ and $\lambda_a^{(j)}(t)$ are

independent when $i \neq j$ [17], [14], we get a recursion for $E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big]$ as

$$
\begin{aligned}
E\Big[\boldsymbol{\lambda}_a(t+1)\boldsymbol{\lambda}_a^T(t+1)\Big] &= E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big] + \mu\mathrm{diag}\Big(\boldsymbol{\gamma}(t)\Big)E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big] \\
&- \mu\mathrm{diag}\Big(\boldsymbol{\Gamma}(t)\bar{\boldsymbol{\lambda}}_a(t)\Big)E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big] \\
&- \mu E\Big[\mathrm{diag}^2(\boldsymbol{u}(t))\Big]\Bigg(E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big] - \bar{\boldsymbol{\lambda}}_a(t)\bar{\boldsymbol{\lambda}}_a^T(t)\Bigg)\mathbf{1}\bar{\boldsymbol{\lambda}}_a^T(t) \\
&- \mu\mathrm{diag}\Big(\bar{\boldsymbol{\lambda}}_a(t)\Big)\boldsymbol{\Gamma}(t)\Bigg(E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big] - \bar{\boldsymbol{\lambda}}_a(t)\bar{\boldsymbol{\lambda}}_a^T(t)\Bigg) \\
&+ \mu E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big]\mathrm{diag}\Big(\boldsymbol{\gamma}(t)\Big) - \mu E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big]\mathrm{diag}\Big(\boldsymbol{\Gamma}(t)\bar{\boldsymbol{\lambda}}_a(t)\Big) \\
&- \mu\bar{\boldsymbol{\lambda}}_a(t)\mathbf{1}^T\Bigg(E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big] - \bar{\boldsymbol{\lambda}}_a(t)\bar{\boldsymbol{\lambda}}_a^T(t)\Bigg)E\Big[\mathrm{diag}^2(\boldsymbol{u}(t))\Big] \\
&- \mu\Bigg(E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big] - \bar{\boldsymbol{\lambda}}_a(t)\bar{\boldsymbol{\lambda}}_a^T(t)\Bigg)\boldsymbol{\Gamma}(t)\mathrm{diag}\Big(\bar{\boldsymbol{\lambda}}_a(t)\Big). \qquad (20)
\end{aligned}
$$

Defining $\boldsymbol{q}_a(t) \triangleq \bar{\boldsymbol{\lambda}}_a(t)$ and $\boldsymbol{Q}_a(t) \triangleq E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big]$, we express (19) and (20) as a coupled recursions in Table 1.

Table 1

Time evolution of the mean and the variance of the affinely constrained mixture weights updated with the EGU algorithm

$$
\begin{aligned}
&\boldsymbol{q}_a(t+1) = \boldsymbol{q}_a(t) + \mu\mathrm{diag}\Big(\boldsymbol{\gamma}(t)\Big)\boldsymbol{q}_a(t) - \mu\mathrm{diag}\Big(\boldsymbol{Q}_a(t)\boldsymbol{\Gamma}(t)\Big), \\
&\boldsymbol{Q}_a(t+1) = \Big(I + \mu\mathrm{diag}\Big(\boldsymbol{\gamma}(t)\Big) - \mu\mathrm{diag}\Big(\boldsymbol{\Gamma}(t)\boldsymbol{q}_a(t)\Big)\Big)\boldsymbol{Q}_a(t) - \mu E\Big[\mathrm{diag}^2(\boldsymbol{u}(t))\Big]\Big(\boldsymbol{Q}_a(t) - \boldsymbol{q}_a(t)\boldsymbol{q}_a^T(t)\Big)\mathbf{1}\boldsymbol{q}_a^T(t) \\
&-\mu\mathrm{diag}\Big(\boldsymbol{q}_a(t)\Big)\boldsymbol{\Gamma}(t)\Big(\boldsymbol{Q}_a(t) - \boldsymbol{q}_a(t)\boldsymbol{q}_a^T(t)\Big) + \boldsymbol{Q}_a(t)\Big(\mu\mathrm{diag}\Big(\boldsymbol{\gamma}(t)\Big) - \mu\mathrm{diag}\Big(\boldsymbol{\Gamma}(t)\boldsymbol{q}_a(t)\Big)\Big) \\
&-\mu\boldsymbol{q}_a(t)\mathbf{1}^T\Big(\boldsymbol{Q}_a(t) - \boldsymbol{q}_a(t)\boldsymbol{q}_a^T(t)\Big)E\Big[\mathrm{diag}^2(\boldsymbol{u}(t))\Big] - \mu\Big(\boldsymbol{Q}_a(t) - \boldsymbol{q}_a(t)\boldsymbol{q}_a^T(t)\Big)\boldsymbol{\Gamma}(t)\mathrm{diag}\Big(\boldsymbol{q}_a(t)\Big).
\end{aligned}
$$

In Table 1, we provide the mean and the variance recursions for $\boldsymbol{Q}_a(t)$ and $\boldsymbol{q}_a(t)$. To implement these recursions, one needs to only provide $\boldsymbol{\Gamma}(t)$ and $\boldsymbol{\gamma}(t)$. Note that $\boldsymbol{\Gamma}(t)$ and $\boldsymbol{\gamma}(t)$ are derived for a wide range of adaptive filters [16], [14]. If we use the mean and the variance recursions in (18), then we obtain the time evolution of the final MSE. This completes the transient analysis of the affinely constrained mixture weights updated with the EGU algorithm.

4.1.2 Relative Entropy

For the affinely constrained combination updated with the EG algorithm, we have the multiplicative updates as

$$\lambda_1^{(i)}(t+1) = u\frac{\lambda_1^{(i)}(t)\exp\{\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\}}{\sum_{k=1}^{m-1}\left[\lambda_1^{(k)}(t)\exp\{\mu e(t)(\hat{y}_k(t)-\hat{y}_m(t))\}+\lambda_2^{(k)}(t)\exp\{-\mu e(t)(\hat{y}_k(t)-\hat{y}_m(t))\}\right]},$$
$$\lambda_2^{(i)}(t+1) = u\frac{\lambda_2^{(i)}(t)\exp\{-\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\}}{\sum_{k=1}^{m-1}\left[\lambda_1^{(k)}(t)\exp\{\mu e(t)(\hat{y}_k(t)-\hat{y}_m(t))\}+\lambda_2^{(k)}(t)\exp\{-\mu e(t)(\hat{y}_k(t)-\hat{y}_m(t))\}\right]},$$

for $i = 1, \ldots, m-1$. Using the same approximations as in (7), (8), (9) and (10), we obtain

$$\lambda_1^{(i)}(t+1) = u\frac{\lambda_1^{(i)}(t)\big(1+\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\big)}{\sum_{k=1}^{m-1}\left[\lambda_1^{(k)}(t)\big(1+\mu e(t)(\hat{y}_k(t)-\hat{y}_m(t))\big)+\lambda_2^{(k)}(t)\big(1-\mu e(t)(\hat{y}_k(t)-\hat{y}_m(t))\big)\right]}, \quad (21)$$
$$\lambda_2^{(i)}(t+1) = u\frac{\lambda_2^{(i)}(t)\big(1-\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\big)}{\sum_{k=1}^{m-1}\left[\lambda_1^{(k)}(t)\big(1+\mu e(t)(\hat{y}_k(t)-\hat{y}_m(t))\big)+\lambda_2^{(k)}(t)\big(1-\mu e(t)(\hat{y}_k(t)-\hat{y}_m(t))\big)\right]}. \quad (22)$$

In our simulations, we illustrate the accuracy of the approximations (21) and (22) under the mixture framework. Using (21) and (22), we obtain updates on $\boldsymbol{\lambda}_1(t)$ and $\boldsymbol{\lambda}_2(t)$ as

$$\boldsymbol{\lambda}_1(t+1) = u\frac{\Big(I+\mu e(t)\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\Big)\boldsymbol{\lambda}_1(t)}{\Big[\mathbf{1}^T+\mu e(t)\boldsymbol{u}^T(t)\Big]\boldsymbol{\lambda}_a(t)}, \quad (23)$$
$$\boldsymbol{\lambda}_2(t+1) = u\frac{\Big(I-\mu e(t)\mathrm{diag}\Big(\boldsymbol{\delta}(t)\Big)\Big)\boldsymbol{\lambda}_2(t)}{\Big[\mathbf{1}^T+\mu e(t)\boldsymbol{u}^T(t)\Big]\boldsymbol{\lambda}_a(t)}. \quad (24)$$

Using updates (23) and (24), we can write update on $\boldsymbol{\lambda}_a(t)$

$$\boldsymbol{\lambda}_a(t+1) = u\frac{\Big[I+\mu e(t)\mathrm{diag}\Big(\boldsymbol{u}(t)\Big)\Big]\boldsymbol{\lambda}_a(t)}{\Big[\mathbf{1}^T+\mu e(t)\boldsymbol{u}^T(t)\Big]\boldsymbol{\lambda}_a(t)}. \quad (25)$$

For the recursion of $\bar{\boldsymbol{\lambda}}_a(t)$, using (25), we get

$$E\Big[\boldsymbol{\lambda}_a(t+1)\Big] = E\left\{u\frac{\Big[I+\mu e(t)\mathrm{diag}\Big(\boldsymbol{u}(t)\Big)\Big]\boldsymbol{\lambda}_a(t)}{\Big[\mathbf{1}^T+\mu e(t)\boldsymbol{u}^T(t)\Big]\boldsymbol{\lambda}_a(t)}\right\},$$

$$\approx u\frac{E\Big\{\Big[I+\mu e(t)\mathrm{diag}\Big(\boldsymbol{u}(t)\Big)\Big]\boldsymbol{\lambda}_a(t)\Big\}}{E\Big\{\Big[\mathbf{1}^T+\mu e(t)\boldsymbol{u}^T(t)\Big]\boldsymbol{\lambda}_a(t)\Big\}}, \tag{26}$$

$$= u\frac{E\Big[\boldsymbol{\lambda}_a(t)\Big]+\mu\mathrm{diag}\Big(\boldsymbol{\gamma}(t)\Big)E\Big[\boldsymbol{\lambda}_a(t)\Big]-\mu\mathrm{diag}\Big(E[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)]\boldsymbol{\Gamma}(t)\Big)}{\Big[\mathbf{1}^T+\mu\boldsymbol{\gamma}^T(t)\Big]E\Big[\boldsymbol{\lambda}_a(t)\Big]-\mu\mathrm{tr}\Big(E[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)]\boldsymbol{\Gamma}(t)\Big)}, \tag{27}$$

where in (26) we approximate expectation of the quotient with the quotient of the expectations. In our simulations, we also illustrate the accuracy of this approximation in the mixture framework. From (25), using the same approximation in (27), assuming $\boldsymbol{\lambda}_a(t)$ is Gaussian, assuming $\lambda_a^{(i)}(t)$ and $\lambda_a^{(j)}(t)$ are independent when $i \neq j$, we get a recursion for $E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big]$ as

$$E\Big[\boldsymbol{\lambda}_a(t+1)\boldsymbol{\lambda}_a^T(t+1)\Big] = u^2\frac{\boldsymbol{A}(t)}{b(t)}, \tag{28}$$

where $\boldsymbol{A}(t)$ is equal to the right hand side of (20) and

$$\begin{aligned}
b(t) &= \mathbf{1}^T E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big]\mathbf{1} + \mu\boldsymbol{p}^T(t)E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big]\mathbf{1} \\
&- \mu\bar{\boldsymbol{\lambda}}_a^T(t)\boldsymbol{R}(t)E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big]\mathbf{1} - \mu\mathbf{1}^T\Big(E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big] - \bar{\boldsymbol{\lambda}}_a(t)\bar{\boldsymbol{\lambda}}_a^T(t)\Big)\boldsymbol{R}(t)\bar{\boldsymbol{\lambda}}_a(t) \\
&- \mu\mathbf{1}^T\Big(E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big] - \bar{\boldsymbol{\lambda}}_a(t)\bar{\boldsymbol{\lambda}}_a^T(t)\Big)E\Big[\mathrm{diag}^2(\boldsymbol{u}(t))\Big]\mathbf{1}^T\bar{\boldsymbol{\lambda}}_a(t)\mathbf{1} \\
&+ \mu\mathbf{1}^T E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big]\boldsymbol{p}(t) - \mu\mathbf{1}^T E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big]\boldsymbol{R}(t)\bar{\boldsymbol{\lambda}}_a(t) \\
&- \mu\bar{\boldsymbol{\lambda}}_a^T(t)\boldsymbol{R}(t)\Big(E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big] - \bar{\boldsymbol{\lambda}}_a(t)\bar{\boldsymbol{\lambda}}_a^T(t)\Big)\mathbf{1} \\
&- \mu\mathbf{1}^T\bar{\boldsymbol{\lambda}}_a^T(t)\mathbf{1}E\Big[\mathrm{diag}^2(\boldsymbol{u}(t))\Big]\Big(E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big] - \bar{\boldsymbol{\lambda}}_a(t)\bar{\boldsymbol{\lambda}}_a^T(t)\Big)\mathbf{1}.
\end{aligned} \tag{29}$$

If we use the mean (27) and the variance (28), (29) recursions in (18), then we obtain the time evolution of the final MSE. This completes the transient analysis of the affinely constrained mixture weights updated with the EG algorithm.

4.2 Unconstrained Mixture

We use the unconstrained relative entropy and the relative entropy as distance measures to update unconstrained mixture weights. We first perform transient analysis of the mixture weights updated using the EGU algorithm. Then, we continue with the transient analysis of the mixture weights updated using the EG algorithm. Note that since the unconstrained case is close to the affinely constrained case, we only provide the necessary modifications to get the mean and the variance recursions for the transient analysis.

4.2.1 Unconstrained Relative Entropy

For the unconstrained combination updated with EGU, we have the multiplicative updates as

$$w_1^{(i)}(t+1) = w_1^{(i)}(t) \exp\left\{\mu e(t)\hat{y}_i(t)\right\},$$
$$w_2^{(i)}(t+1) = w_2^{(i)}(t) \exp\left\{-\mu e(t)\hat{y}_i(t)\right\},$$

for $i = 1, \ldots, m$. Using the same approximations as in (7), (8), (9) and (10), we can obtain updates on $\boldsymbol{w}_1(t)$ and $\boldsymbol{w}_2(t)$ as

$$\boldsymbol{w}_1(t+1) = \Big(I + \mu e(t)\mathrm{diag}\Big(\boldsymbol{x}(t)\Big)\Big)\boldsymbol{w}_1(t), \tag{30}$$
$$\boldsymbol{w}_2(t+1) = \Big(I - \mu e(t)\mathrm{diag}\Big(\boldsymbol{x}(t)\Big)\Big)\boldsymbol{w}_2(t). \tag{31}$$

Collecting the weights in $\boldsymbol{w}_a(t) = [\boldsymbol{w}_1(t); \boldsymbol{w}_2(t)]$, using the updates (30) and (31), we can write update on $\boldsymbol{w}_a(t)$ as

$$\boldsymbol{w}_a(t+1) = \Big(I + \mu e(t)\mathrm{diag}\Big(\boldsymbol{u}(t)\Big)\Big)\boldsymbol{w}_a(t), \tag{32}$$

where $\boldsymbol{u}(t)$ is defined as $\boldsymbol{u}(t) \stackrel{\triangle}{=} [\boldsymbol{x}(t); -\boldsymbol{x}(t)]$.

For the desired signal $y(t)$, we can write $y(t) = \boldsymbol{w}_0^T(t)\boldsymbol{x}(t) + e_0(t)$, where $\boldsymbol{w}_0(t)$ is the optimum MSE solution at time t such that $\boldsymbol{w}_0(t) \stackrel{\triangle}{=} \boldsymbol{R}^{-1}(t)\boldsymbol{p}(t)$,

$\boldsymbol{R}(t) \triangleq E\Big[\boldsymbol{x}(t)\boldsymbol{x}^T(t)\Big]$, $\boldsymbol{p}(t) \triangleq E\left\{\boldsymbol{x}(t)y(t)\right\}$ and $e_0(t)$ is zero-mean disturbance uncorrelated to $\boldsymbol{x}(t)$. To show that the mixture weights converge to the optimum solution in the steady-state such that $\lim_{t\to\infty} E\Big[\boldsymbol{w}(t)\Big] = \lim_{t\to\infty} \boldsymbol{w}_0(t)$, we follow similar lines as in the Section 4.1.1. We modify (14), (15), (16) and (17) such that $\boldsymbol{\lambda}$ will be replaced by $\boldsymbol{w}$, $\boldsymbol{\delta}(t)$ will be replaced by $\boldsymbol{x}(t)$ and $\boldsymbol{\varepsilon}(t) = \boldsymbol{w}_0(t) - \boldsymbol{w}(t)$. After these replacements, we obtain

$$\begin{aligned} E\Big[\boldsymbol{\varepsilon}(t+1)\Big] &= E\Big[I - \mu\mathrm{diag}\Big(\boldsymbol{w}_1(t) + \boldsymbol{w}_2(t)\Big)\boldsymbol{x}(t)\boldsymbol{x}^T(t)\Big]E\Big[\boldsymbol{\varepsilon}(t)\Big] \\ &+ E\Big[\boldsymbol{w}_0(t+1) - \boldsymbol{w}_0(t)\Big]. \end{aligned} \tag{33}$$

Since, we have $\lim_{t\to\infty} E\Big[\boldsymbol{w}_0(t+1) - \boldsymbol{w}_0(t)\Big] = 0$ for most adaptive filters in the first stage [14] and if μ is chosen so that all the eigenvalues of $E\Big[I - \mu\mathrm{diag}\Big(\boldsymbol{w}_1(t) + \boldsymbol{w}_2(t)\Big)\boldsymbol{x}(t)\boldsymbol{x}^T(t)\Big]$ have strictly less than unit magnitude for every t, then $\lim_{t\to\infty} E\Big[\boldsymbol{w}(t)\Big] = \lim_{t\to\infty} \boldsymbol{w}_0(t)$.

For the transient analysis of MSE, defining $\boldsymbol{\gamma}(t) \triangleq E\left\{\boldsymbol{u}(t)y(t)\right\}$ and $\boldsymbol{\Gamma}(t) \triangleq E\Big[\boldsymbol{u}(t)\boldsymbol{u}^T(t)\Big]$, (18) is modified as

$$E[e^2(t)] = E\left\{y^2(t)\right\} - 2\bar{\boldsymbol{w}}_a^T(t)\boldsymbol{\gamma}(t) + \mathrm{tr}\left(E\left[\boldsymbol{w}_a(t)\boldsymbol{w}_a^T(t)\right]\boldsymbol{\Gamma}(t)\right). \tag{34}$$

Accordingly, we modify the mean recursion (19) and the variance recursion (20) such that instead of $\boldsymbol{\lambda}_a(t)$ we use $\boldsymbol{w}_a(t)$. We also modify the Table 1 using $\boldsymbol{q}_a(t) \triangleq \bar{\boldsymbol{w}}_a(t)$ and $\boldsymbol{Q}_a(t) \triangleq E\Big[\boldsymbol{w}_a(t)\boldsymbol{w}_a^T(t)\Big]$. If we use this modified mean and variance recursions in (34), then we obtain the time evolution of the final MSE. This completes the transient analysis of the unconstrained mixture weights updated with the EGU algorithm.

4.2.2 Relative Entropy

For the unconstrained combination updated with the EG algorithm, we have the multiplicative updates as

$$w_a^{(i)}(t+1) = u\frac{w_a^{(i)}(t)\exp\{\mu e(t)\hat{y}_i(t)\}}{\sum_{k=1}^{m}\left[w_a^{(k)}(t)\exp\{\mu e(t)\hat{y}_k(t)\} + w_a^{(k+m)}(t)\exp\{-\mu e(t)\hat{y}_k(t)\}\right]},$$

$i = 1, \ldots, m,$

$$w_a^{(i)}(t+1) = u\frac{w_a^{(i)}(t)\exp\{-\mu e(t)\hat{y}_i(t)\}}{\sum_{k=1}^{m}\left[w_a^{(k)}(t)\exp\{\mu e(t)\hat{y}_k(t)\} + w_a^{(k+m)}(t)\exp\{-\mu e(t)\hat{y}_k(t)\}\right]},$$

$i = m+1 \ldots, 2m.$

Following similar lines, we modify (23), (24), (25), (27), (28) and (29) such that we replace $\boldsymbol{\delta}(t)$ with $\boldsymbol{x}(t)$, $\boldsymbol{\lambda}$ with $\boldsymbol{w}$ and $\boldsymbol{u}(t) = \left[\boldsymbol{x}(t); -\boldsymbol{x}(t)\right]$. Finally, we use the modified mean and variance recursions in (34) and obtain the time evolution of the final MSE. This completes the transient analysis of the unconstrained mixture weights updated with the EG algorithm.

5 Simulations

In this section, we illustrate the accuracy of our results and compare performances of different adaptive mixture methods through simulations. In our simulations, we observe that using the EG algorithm to train the mixture weights yields better performance compared to using the LMS algorithm or the EGU algorithm to train the mixture weights for combinations having more than two filters and when the combination favors only a few of the constituent filters. The LMS algorithm and the EGU algorithm perform similarly in our simulations when they are used to train the mixture weights. We also observe in our simulations that the mixture weights under the EG update converge to the optimum combination vector faster than the mixture weights under the

LMS algorithm.

To compare performances of the EG and LMS algorithms and illustrate the accuracy of our results in (27), (28) and (29) under different algorithmic parameters, the desired signal as well as the system parameters are selected as follows. First, a seventh-order linear filter,
$\boldsymbol{w}_o = [0.25, -0.47, -0.37, 0.045, -0.18, 0.78, 0.147]^T$, is chosen as in [17]. The underlying signal is generated using the data model $y(t) = \tau\, \boldsymbol{w}_o^T \boldsymbol{a}(t) + n(t)$, where $\boldsymbol{a}(t)$ is an i.i.d. Gaussian vector process with zero mean and unit variance entries, i.e., $E[\boldsymbol{a}(t)\boldsymbol{a}^T(t)] = \boldsymbol{I}$, $n(t)$ is an i.i.d. Gaussian noise process with zero mean and variance $E[n^2(t)] = 0.3$, and τ is a positive scalar to control SNR. Hence, the SNR of the desired signal is given by SNR $\triangleq$ $10\log(\frac{E[\tau^2(\boldsymbol{w}_o^T\boldsymbol{u}(t))^2]}{0.01}) = 10\log(\frac{\tau^2\|\boldsymbol{w}_o\|^2}{0.01})$. For the first experiment, we have SNR = -10dB. To model the unknown system we use ten linear filters using the LMS update as the constituent filters. The learning rates of these two constituent filters are set to $\mu_1 = 0.002$ and $\mu_6 = 0.002$ while the learning rates for the rest of the constituent filters are selected randomly in $[0.1, 0.11]$. Therefore, in the steady-state, we obtain the optimum combination vector approximately as $\boldsymbol{\lambda}_o = [0.5, 0, 0, 0, 0, 0.5, 0, 0, 0, 0]^T$, i.e., the final combination vector is sparse. In the second stage, we train the combination weights with the EG and LMS algorithms and compare performances of these algorithms. For the second stage, the learning rates for the EG and LMS algorithms are selected as $\mu_{\text{EG}} = 0.0008$ and $\mu_{\text{LMS}} = 0.005$ such that the MSEs of both mixtures converge to the same final MSE to provide a fair comparison. We select $u = 500$ for the EG algorithm. In Fig. 2a, we plot the weight of the first constituent filter with $\mu_1 = 0.002$, i.e. $E[\boldsymbol{\lambda}^{(1)}(t)]$, updated with the EG and LMS algorithms. In Fig. 2b, we plot the MSE curves for the adaptive mixture updated with the EG algorithm, the adaptive mixture updated with the LMS algorithm, the first constituent filter with $\mu_1 = 0.002$ and the second constituent filter with $\mu_2 \in [0.1, 0.11]$. From Fig. 2a and Fig. 2b, we see

that the EG algorithm performs better than the LMS algorithm such that the combination weight under the update of the EG algorithm converges to 0.5 faster than the combination weight under the update of the LMS algorithm. Furthermore the MSE of the adaptive mixture updated with the EG algorithm converges faster than the MSE of the adaptive mixture updated with the LMS algorithm. In Fig. 2c, to test the accuracy of (27), we plot the theoretical values for $\bar{\lambda}_a^{(1)}(t)$ and $\bar{\lambda}_a^{(10)}(t)$ along with simulations. Note in Fig. 2c we observe that $\bar{\lambda}^{(1)}(t) = \bar{\lambda}_a^{(1)}(t) - \bar{\lambda}_a^{(10)}(t)$ converges to 0.5 as predicted in our derivations. In Fig. 2d, to test the accuracy of (28) and (29), as an example, we plot the theoretical values of $E\Big[\lambda_a^{(1)}(t)^2\Big]$ and $E\Big[\lambda_a^{(1)}(t)\lambda_a^{(3)}(t)\Big]$ along with simulations. As we observe from Fig. 2c and Fig. 2d, there is a close agreement between our results and simulations in these experiments. We observe similar results for the other cross terms.

We next simulate the unconstrained mixtures updated with the EGU and EG algorithms. Here, we have two linear filters and both using the LMS update to train their weight vectors as the constituent filters. The learning rates for two constituent filters are set to $\mu_1 = 0.002$ and $\mu_2 = 0.1$ respectively. Therefore, in the steady-state, we obtain the optimum vector approximately as $\boldsymbol{w}_o = [1, 0]$. We have SNR = 1 for these simulations. The unconstrained mixture weights are first updated with the EGU algorithm. For the second stage, the learning rate for the EGU algorithm is selected as $\mu_{\text{EGU}} = 0.01$. The theoretical curves in the figures are produced using $\boldsymbol{\Gamma}(t)$ and $\boldsymbol{\gamma}(t)$ that are calculated from the simulations, since our goal is to illustrate the validity of derived equations. In Fig. 3a, we plot the theoretical values of $\bar{\boldsymbol{w}}_a^{(1)}(t)$, $\bar{\boldsymbol{w}}_a^{(2)}(t)$, $\bar{\boldsymbol{w}}_a^{(3)}(t)$ and $\bar{\boldsymbol{w}}_a^{(4)}(t)$ along with simulations. In Fig. 3b, as an example, we plot the theoretical values of $E\Big[\boldsymbol{w}_a^{(1)}(t)^2\Big]$, $E\Big[\boldsymbol{w}_a^{(1)}(t)\boldsymbol{w}_a^{(2)}(t)\Big]$, $E\Big[\boldsymbol{w}_a^{(2)}(t)\boldsymbol{w}_a^{(3)}(t)\Big]$ and $E\Big[\boldsymbol{w}_a^{(3)}(t)\boldsymbol{w}_a^{(4)}(t)\Big]$ along with simulations. We continue to update the mixture weights with the EG algorithm. For the second stage, the learning rate for the EG algorithm is selected as $\mu_{\text{EG}} = 0.01$. We select $u = 3$ for the EG algorithm. In Fig. 3c,

we plot the theoretical values of $\bar{\boldsymbol{w}}_a^{(1)}(t)$, $\bar{\boldsymbol{w}}_a^{(2)}(t)$, $\bar{\boldsymbol{w}}_a^{(3)}(t)$ and $\bar{\boldsymbol{w}}_a^{(4)}(t)$ along with simulations. In Fig. 3d, as an example, we plot the theoretical values of $E\Big[\boldsymbol{w}_a^{(2)}(t)^2\Big]$, $E\Big[\boldsymbol{w}_a^{(1)}(t)\boldsymbol{w}_a^{(2)}(t)\Big]$, $E\Big[\boldsymbol{w}_a^{(2)}(t)\boldsymbol{w}_a^{(3)}(t)\Big]$ and $E\Big[\boldsymbol{w}_a^{(2)}(t)\boldsymbol{w}_a^{(4)}(t)\Big]$ along with simulations. We observe a close agreement between our results and simulations.

To test the accurateness of the assumptions in (9) and (10), we plot in Fig. 4a, the difference

$$\frac{\| \exp\{\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\} - \{1+\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\} \|^2}{\sqrt{\| \exp\{\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\} \|^2 \| \{1+\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\} \|^2}}$$

for $i = 1$ with the same algorithmic parameters as in Fig. 2 and Fig. 3. To test the accurateness of the separation assumption in (27), we plot in Fig. 4b, the first parameter of the difference

$$\frac{\left\| E\left\{ u \frac{\left[I+\mu e(t)\mathrm{diag}\left(\boldsymbol{u}(t)\right)\right]\boldsymbol{\lambda}_a(t)}{\left[\mathbf{1}^T+\mu e(t)\boldsymbol{u}^T(t)\right]\boldsymbol{\lambda}_a(t)} \right\} - u \frac{E\left\{ \left[I+\mu e(t)\mathrm{diag}\left(\boldsymbol{u}(t)\right)\right]\boldsymbol{\lambda}_a(t) \right\}}{E\left\{ \left[\mathbf{1}^T+\mu e(t)\boldsymbol{u}^T(t)\right]\boldsymbol{\lambda}_a(t) \right\}} \right\|^2}{\sqrt{\left\| E\left\{ u \frac{\left[I+\mu e(t)\mathrm{diag}\left(\boldsymbol{u}(t)\right)\right]\boldsymbol{\lambda}_a(t)}{\left[\mathbf{1}^T+\mu e(t)\boldsymbol{u}^T(t)\right]\boldsymbol{\lambda}_a(t)} \right\} \right\|^2 \left\| u \frac{E\left\{ \left[I+\mu e(t)\mathrm{diag}\left(\boldsymbol{u}(t)\right)\right]\boldsymbol{\lambda}_a(t) \right\}}{E\left\{ \left[\mathbf{1}^T+\mu e(t)\boldsymbol{u}^T(t)\right]\boldsymbol{\lambda}_a(t) \right\}} \right\|^2}}$$

with the same algorithmic parameters as in Fig. 2 and Fig. 3. We observe that assumptions are fairly accurate for these algorithms in our simulations.

In the last simulations, we compare performances of the EGU, EG and LMS algorithms updating the affinely mixture weights under different algorithmic parameters. Algorithmic parameters and constituent filters are selected as in Fig. 2 under SNR = -5 and 5. For the second stage, under SNR = -5, learning rates for the EG, EGU and LMS algorithms are selected as $\mu_{\mathrm{EG}} = 0.0005$, $\mu_{\mathrm{EGU}} = 0.005$ and $\mu_{\mathrm{LMS}} = 0.005$ such that the MSEs converge to the same final MSE to provide a fair comparison. We choose u = 500 for the EG algorithm. In Fig. 5a, we plot the MSE curves for the adaptive mixture updated with the EG algorithm, the adaptive mixture updated with the EGU algorithm, the adaptive mixture updated with the LMS algorithm, first constituent filter

with $\mu_1 = 0.002$ and second constituent filter with $\mu_2 \in [0.1, 0.11]$ under SNR = -5. Under SNR = 5, learning rates for the EG, EGU and LMS algorithms are selected as $\mu_{\text{EG}} = 0.002$, $\mu_{\text{EGU}} = 0.005$ and $\mu_{\text{LMS}} = 0.005$. We choose u = 100 for the EG algorithm. In Fig. 5b, we plot same MSE curves as in Fig. 5a. We observe that the EG algorithm performs better than the EGU and LMS algorithms such that MSE of the adaptive mixture updated with the EG algorithm converges faster than the MSE of adaptive mixtures updated with the EGU and LMS algorithms. We also observe that the EGU and LMS algorithms show similar performances when they are used to train the mixture weights.

6 Conclusion

In this paper, we investigate adaptive mixture methods based on Bregman divergences combining outputs of m adaptive filters to model a desired signal. We use the unnormalized relative entropy and relative entropy as distance measures that produce the exponentiated gradient update with unnormalized weights (EGU) and the exponentiated gradient update with positive and negative weights (EG) to train the mixture weights under the affine constraints or without any constraints. We provide the transient analysis of these methods updated with the EGU and EG algorithms. In our simulations, we compare performances of the EG, EGU and LMS algorithms and observe that the EG algorithm performs better than the EGU and LMS algorithms when the combination vector in steady-state is sparse. We observe that the EGU and LMS algorithms show similar performance when they are used to train the mixture weights. We also observe a close agreement between the simulations and our theoretical results.

References

[1] C. Boukis, D. Mandic, A. G. Constantinides, A class of stochastic gradient algorithms with exponentiated error cost functions, Digital Signal Processing 19 (2009) 201–212.

[2] D. P. Helmbold, R. E. Schapire, Y. Singer, M. K. Warmuth, A comparison of new and old algorithms for a mixture estimation problem, Machine Learning 27 (1997) 97–119.

[3] J. C. M. Bermudez, N. J. Bershad, J. Y. Tourneret, Stochastic analysis of an error power ratio scheme applied to the affine combination of two lms adaptive filters, Signal Processing 91 (2011) 2615–2622.

[4] J. Arenas-Garcia, M. Martinez-Ramon, A. Navia-Vazquez, A. R. Figueiras-Vidal, Plant identification via adaptive combination of transversal filters, Signal Processing 86 (2006) 2430–2438.

[5] S. S. Kozat, A. C. Singer, Multi-stage adaptive signal processing algorithms, in: Proceedings of SAM Signal Proc. Workshop, 2000, pp. 380–384.

[6] J. Arenas-Garcia, V. Gomez-Verdejo, M. Martinez-Ramon, A. R. Figueiras-Vidal, Separate-variable adaptive combination of LMS adaptive filters for plant identification, in: Proc. of the 13th IEEE Int. Workshop Neural Networks Signal Processing, 2003, pp. 239–248.

[7] J. Arenas-Garcia, M. Martinez-Ramon, V. Gomez-Verdejo, A. R. Figueiras-Vidal, Multiple plant identifier via adaptive LMS convex combination, in: Proc. of the IEEE Int. Symp. Intel. Signal Processing, 2003, pp. 137–142.

[8] J. Arenas-Garcia, V. Gomez-Verdejo, A. R. Figueiras-Vidal, New algorithms for improved adaptive convex combination of lms transversal filters, IEEE Transactions on Instrumentation and Measurement 54 (2005) 2239–2249.

[9] J. Kivinen, M. Warmuth, Exponentiated gradient versus gradient descent for linear predictors 132 (1997) 1–64.

[10] D. P. Helmbold, R. E. Schapire, Y. Singer, M. K. Warmuth, On-line portfolio selection using multiplicative updates, Mathematical Finance 8 (4) (1998) 325347.

[11] N. J. Bershad, J. C. M. Bermudez, J. Tourneret, An affine combination of two LMS adaptive filters: Transient mean-square analysis, IEEE Transactions on Signal Processing 56 (5) (2008) 1853–1864.

[12] S. S. Kozat, A. T. Erdogan, A. C. Singer, A. H. Sayed, Steady state MSE performance analysis of mixture approaches to adaptive filtering, IEEE Transactions on Signal Processing 58 (2010) 4421–4427.

[13] J. Benesty, Y. A. Huang, The LMS, PNLMS, and Exponentiated Gradient algorithms, Proc. Eur. Signal Process. Conf. (EUSIPCO) (2004) 721–724.

[14] A. H. Sayed, Fundamentals of Adaptive Filtering, John Wiley and Sons, 2003.

[15] J. Arenas-Garcia, A. R. Figueiras-Vidal, A. H. Sayed, Mean-square performance of a convex combination of two adaptive filters, IEEE Transactions on Signal Processing 54 (2006) 1078–1090.

[16] V. H. Nascimento, M. T. M. Silva, J. Arenas-Garcia, A transient analysis for the convex combination of adaptive filters, IEEE Transactions on Signal Processing 58 (8) (2009) 4064–4078.

[17] S. S. Kozat, A. T. Erdogan, A. C. Singer, A. H. Sayed, Transient analysis of adaptive affine combinations, accepted, 2011.

[18] V. Vovk, A game of prediction with expert advice, Journal of Computer and System Sciences 56 (1998) 153–173.

[19] P. A. Naylor, J. Cui, M. Brookes, Adaptive algorithms for sparse echo cancellation, Signal Processing 86 (2006) 1182–1192.

[20] N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, M. K. Warmuth, How to use expert advice, Journal of the ACM 44 (3) (1997) 427–485.

[21] B. Jelfs, D. P. Mandic, S. C. Douglas, An adaptive approach for the identification of improper complex signals, Signal Processing 92 (2012) 335–344.

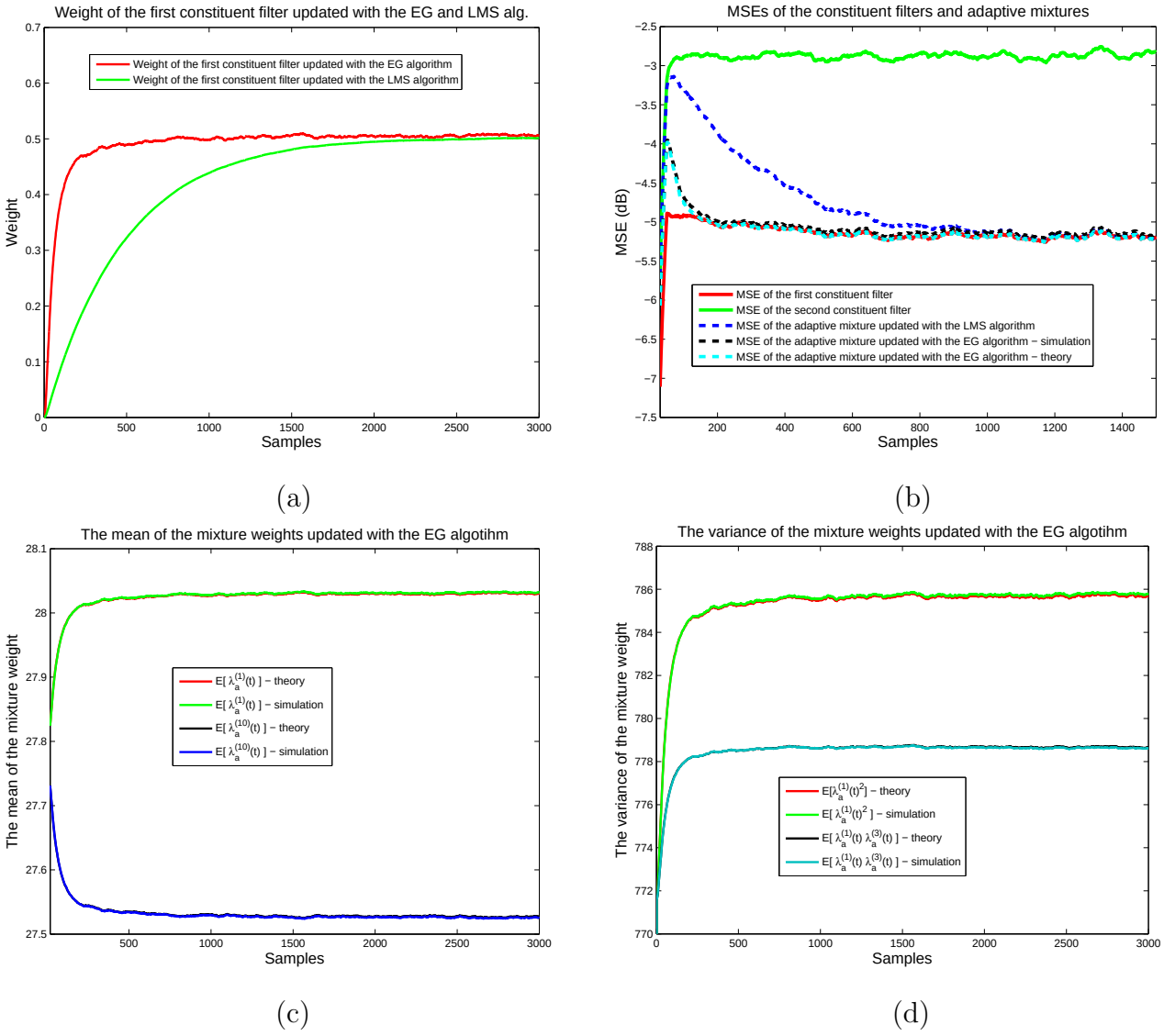


Fig. 2. Using 10 LMS filters as constituent filters, where learning rates for 2 constituent filters are $\mu = 0.002$ and for the rest are $\mu \in [0.1, 0.11]$. SNR = -10dB. For the mixture stage, the EG algorithm has $\mu_{\text{EG}} = 0.0008$ and the LMS algorithm has $\mu_{\text{LMS}} = 0.005$. For the EG algorithm, $u = 500$. (a) The weight of the first constituent filter in the mixture, i.e., $E[\boldsymbol{\lambda}^{(1)}(t)]$. (b) The MSE curves for adaptive mixture updated with the EG algorithm, the adaptive mixture updated with the LMS algorithm, the first constituent filter and the second constituent filter. (c) Theoretical values $\bar{\lambda}_a^{(1)}(t)$ and $\bar{\lambda}_a^{(10)}(t)$ and simulations. (d) Theoretical values $E\big[\lambda_a^{(1)}(t)^2\big]$ and $E\big[\lambda_a^{(1)}(t)\lambda_a^{(3)}(t)\big]$ and simulations.

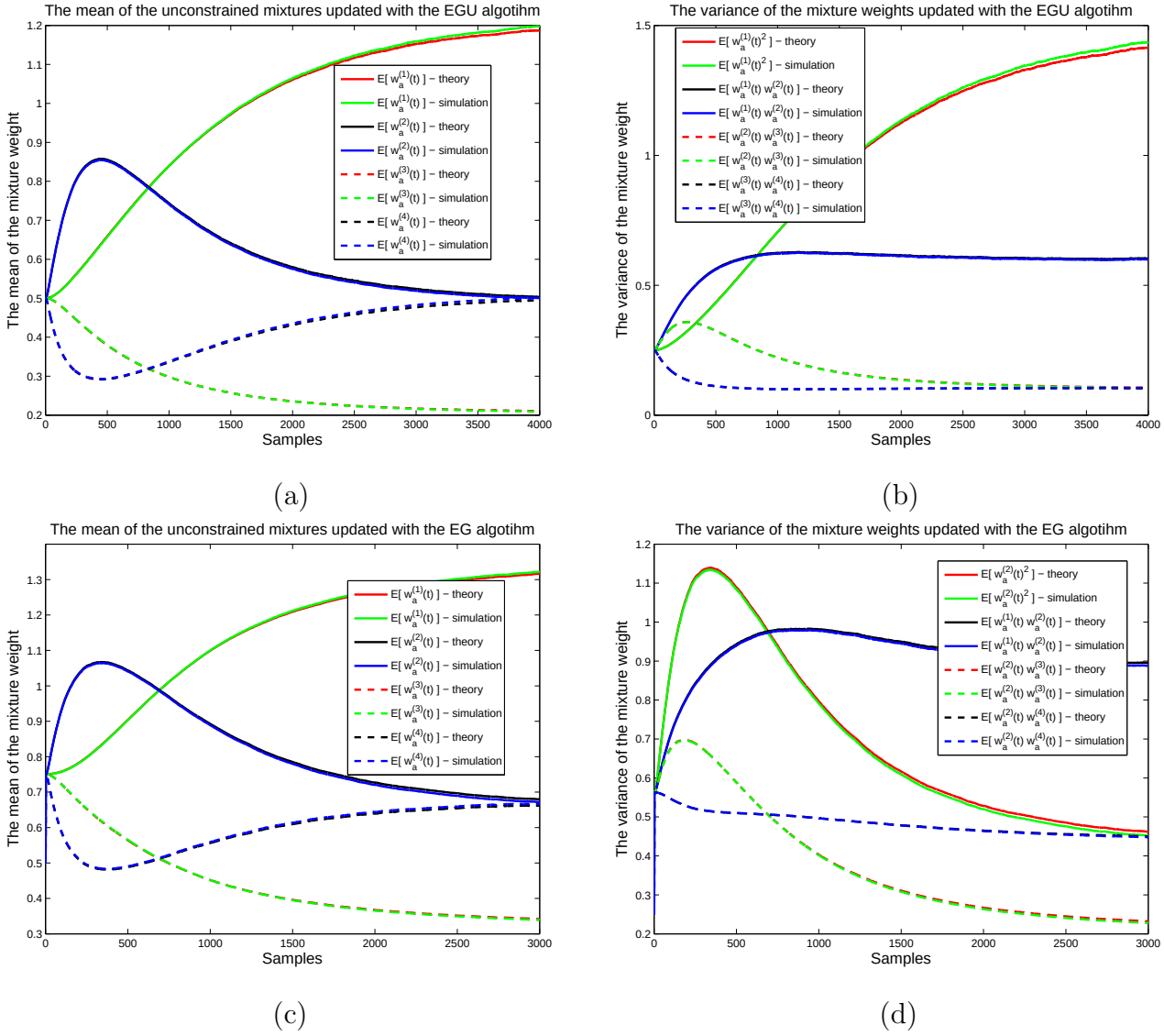
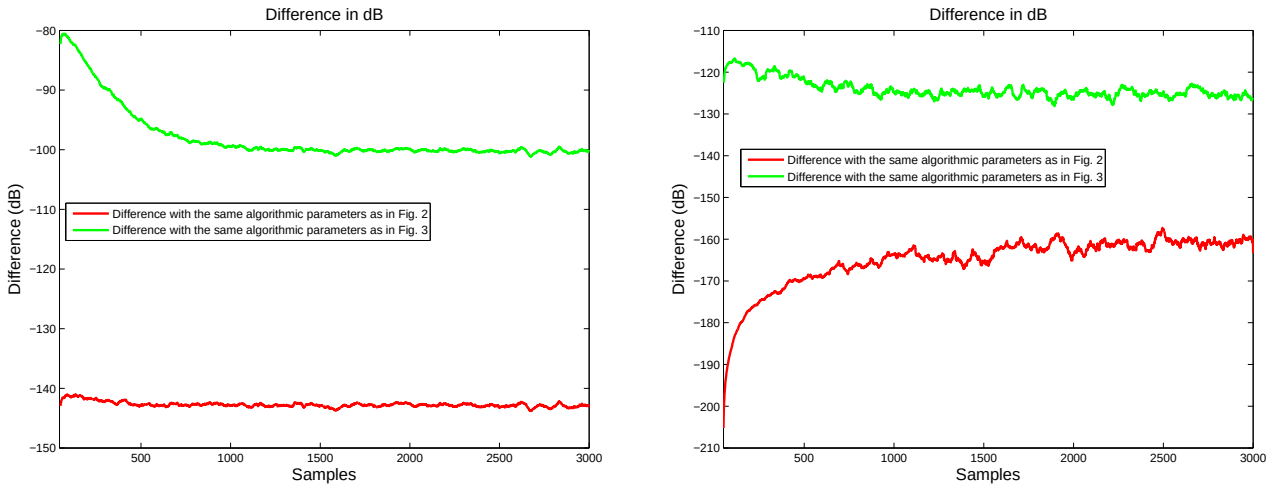
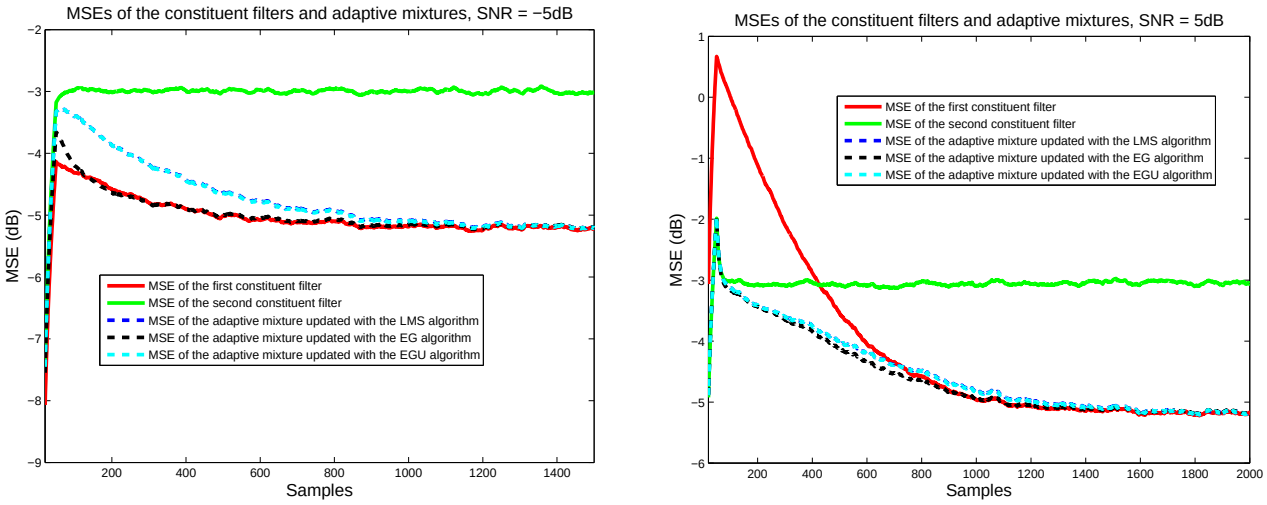


Fig. 3. Two LMS filters as constituent filters with learning rates $\mu_1 = 0.002$ and $\mu_2 = 0.1$, respectively. SNR = 1dB. For the second stage, the EGU algorithm has $\mu_{\text{EGU}} = 0.01$ and the EG algorithm has $\mu_{\text{EG}} = 0.01$. For the EG algorithm, $u = 3$. (a) Theoretical values for the mixture weights updated with the EGU algorithm and simulations. (b) Theoretical values $E\big[\boldsymbol{w}_a^{(1)}(t)^2\big]$, $E\big[\boldsymbol{w}_a^{(1)}(t)\boldsymbol{w}_a^{(2)}(t)\big]$, $E\big[\boldsymbol{w}_a^{(2)}(t)\boldsymbol{w}_a^{(3)}(t)\big]$ and $E\big[\boldsymbol{w}_a^{(3)}(t)\boldsymbol{w}_a^{(4)}(t)\big]$ and simulations. (c) Theoretical mixture weights updated with the EG algorithm and simulations. (d) Theoretical values $E\big[\boldsymbol{w}_a^{(2)}(t)^2\big]$, $E\big[\boldsymbol{w}_a^{(1)}(t)\boldsymbol{w}_a^{(2)}(t)\big]$, $E\big[\boldsymbol{w}_a^{(2)}(t)\boldsymbol{w}_a^{(3)}(t)\big]$ and $E\big[\boldsymbol{w}_a^{(2)}(t)\boldsymbol{w}_a^{(4)}(t)\big]$ and simulations.



(a) (b)

Fig. 4. (a) The difference $\frac{\|\exp\{\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\}-\{1+\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\}\|^2}{\sqrt{\|\exp\{\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\}\|^2\|\{1+\mu e(t)(\hat{y}_i(t)-\hat{y}_m(t))\}\|^2}}$ for $i=1$ with the same algorithmic parameters as in Fig. 2 and Fig. 3. (b) The first parameter of the difference $\frac{\left\|E\left\{u\frac{\left[I+\mu e(t)\mathrm{diag}\left(\boldsymbol{u}_{(t)}\right)\right]\boldsymbol{\lambda}_{a(t)}}{\left[\mathbf{1}^T+\mu e(t)\boldsymbol{u}^T{}_{(t)}\right]\boldsymbol{\lambda}_{a(t)}}\right\}-u\frac{E\left\{\left[I+\mu e(t)\mathrm{diag}\left(\boldsymbol{u}_{(t)}\right)\right]\boldsymbol{\lambda}_{a(t)}\right\}}{E\left\{\left[\mathbf{1}^T+\mu e(t)\boldsymbol{u}^T{}_{(t)}\right]\boldsymbol{\lambda}_{a(t)}\right\}}\right\|^2}{\sqrt{\left\|E\left\{u\frac{\left[I+\mu e(t)\mathrm{diag}\left(\boldsymbol{u}_{(t)}\right)\right]\boldsymbol{\lambda}_{a(t)}}{\left[\mathbf{1}^T+\mu e(t)\boldsymbol{u}^T{}_{(t)}\right]\boldsymbol{\lambda}_{a(t)}}\right\}\right\|^2\left\|u\frac{E\left\{\left[I+\mu e(t)\mathrm{diag}\left(\boldsymbol{u}_{(t)}\right)\right]\boldsymbol{\lambda}_{a(t)}\right\}}{E\left\{\left[\mathbf{1}^T+\mu e(t)\boldsymbol{u}^T{}_{(t)}\right]\boldsymbol{\lambda}_{a(t)}\right\}}\right\|^2}}$ with the same algorithmic parameters as in Fig. 2 and Fig. 3.



(a) (b)

Fig. 5. Algorithmic parameters and constituent filters are selected as in Fig. 2 under SNR = -5dB. For the second stage, the EG algorithm has $\mu_{\mathrm{EG}} = 0.0005$, the EGU algorithm has $\mu_{\mathrm{EGU}} = 0.005$ and the LMS algorithm has $\mu_{\mathrm{LMS}} = 0.005$. For the EG algorithm, u = 500. (a) the MSE curves for the adaptive mixture updated with the EG algorithm, the adaptive mixture updated with the EGU algorithm, the adaptive mixture updated with the LMS algorithm, the first constituent filter and the second constituent filter. Next, SNR = 5dB. For the second stage, the EG algorithm has $\mu_{\mathrm{EG}} = 0.002$, the EGU algorithm has $\mu_{\mathrm{EGU}} = 0.005$ and the LMS algorithm has $\mu_{\mathrm{LMS}} = 0.005$. For the EG algorithm, u = 100. (b) the MSE curves for the adaptive mixture updated with the EG algorithm, the adaptive mixture updated with the EGU algorithm, the adaptive mixture updated with the LMS algorithm, the first constituent filter and the second constituent filter.